iXs Research Corp.
Cool Robotics

iMCs04 ソフトウェアマニュアルVer1.4

株式会社イクシスリサーチ

目　次

１．概要

iMCs04(H8/USB モータコントローラ)は USB 接続によるモータコントロール専用の超小型ボードです．同時に最大 32 個の iMCs04 を接続することが可能です．

本ボードの特徴

- 超小型(45mm x 30mm)
- USB で PC から直接制御が可能.
- 1 枚のボードで最大 4 個のサーボモータが制御可能
- 制御周期は 20ms
- 市販の HUB を介すことで最大 32 枚を同時接続可能
- あらゆるモータ制御が出来るように，コントローラとモータドライバが分離

本マニュアルでは iMCs04 を扱う上で必要な知識を示し,Linux を使用したサンプルを記しています.ハードウェアに関しては,ハードウェアマニュアルをご参照ください.

２．　エンドポイント

２．１　エンドポイント２

エンドポイント２は iMCs04 の各出力ピンから出力されるデューティ比を設定する目的で使用されます．デューティ比は 0/50000(0.0%)～32767/50000(65.536%)までの範囲で設定可能です．

エンドポイント２はバルク転送で１パケットの大きさは 64 バイト，方向は OUT（PC→H8）です．パケットの中身は Fig.1 の構造体で表されます．Short のバイトオーダはリトルエンディアン（インテルなど）です．従ってインテル系の CPU では，そのまま扱うことが出来ます．H8 内部はビッグエンディアンですが，転送前にリトルエンディアンに変換しています．

```
struct ccmd {
        unsigned short duty[4];  // デューティ比設定用(0～32767)
        char dumy[56];
};
```

Fig.1 エンドポイント２の ccmd 構造体

２．２　エンドポイント３

エンドポイント３は iMCs04 の各入力ピンから A/D 変換を行った値，また iMCs04 の ID を取得する目的で使用されます．

エンドポイント３はバルク転送で１パケットの大きさは 64 バイト，方向は IN（H8→PC）です．パケットの中身は次の構造体で表されます．uin 構造体は Fig.2 のようになっています．

```
struct uin {
```



```
	unsigned short time     // 1ms 周期のカウンターの値(0～2047)
	unsigned short magicno; // デバイス ID
	unsigned short ad[4];   // A/D コンバータの値(10bit)
	char dumy[52];
};
```

Fig.2 エンドポイント３の uin 構造体

３　USBドライバについて

３．１　デバイスのオープン，クローズ

USB デバイスをオープンするためには，プログラム内で

```
int fd;
fd = open( “/dev/urbtc0” ,O_RDWR);
```

と記述します．デバイスのオープンに失敗したときは，戻り値-1 が返ります．クローズするためには

```
close(fd)
```

とします．

プログラムは以下のようになります．

```
/************************************************************
  char *dev = "/dev/urbtc0";
  if (argc>1) dev = argv[1];
  if ((fd = open(dev, O_RDWR)) == -1) {
    fprintf(stderr, "%s: Open error¥n", dev);
    exit(1);
  }
************************************************************/
```

３．２　iMCs04 のデータを取り込む

iMCs04 からのデータを連続して取り込むには，まず

```
ioctl(fd, URCC_GET_DATA);
```

を実行しておく必要があります．（一度実行すれば，変更があるまで有効）その後，

```
read(fd, &buf, sizeof(buf));
```

で値を取得します．ここで buf はユーザー定義のエンドポイント３の uin 型構造体です．

プログラムは以下のようになります．

```
/************************************************************
  struct uin buf;
  if (ioctl(fd, URCC_GET_DATA) < 0){
      fprintf(stderr, "ioctl: URCC_GET_DATA error¥n");
      exit(1);
  }
  if ((i = read(fd, &buf, sizeof(buf))) != 64) {
      fprintf(stderr, "Warning: read size mismatch (%d!=%d).¥n", i,sizeof(buf));
      continue;
  }
```



```
**********************************************************/
```

３．３　iMCs04 に制御データを書き込む

iMCs04 に制御データ（エンドポイント２の ccmd）を書き込むには，まず

```
ioctl(fd, URCC_DUTY_SET);
```

を実行しておく必要があります．（一度実行すれば，変更があるまで有効）その後，

```
write(fd, &obuf, sizeof(obuf));
```

で値を書き込みます．ここで obuf はユーザー定義のエンドポイント２の ccmd 型構造体です．

プログラムは以下のようになります．

```
/**********************************************************
  struct ccmd obuf;
  if (ioctl(fd, URCC_DUTY_SET) < 0){
     fprintf(stderr, "ioctl: URCC_DUTY_SET error¥n");
     exit(1);
  }
  if (write(fd, &obuf, sizeof(obuf)) < 0) {
     fprintf(stderr, "write error¥n");
     exit(1);
  }
**********************************************************/
```

３．４　LITTLE_ENDIAN，BIG_ENDIANについて

コンピュータは 2 バイト以上のデータを扱う際に 1 バイトごとに分割して処理しますが，これを最下位のバイトから順番に記録/送信する方式をリトルエンディアン(LITTLE ENDIAN)と呼び，最上位のバイトから順番に記録/送信する方式をビッグエンディアン(BIG ENDIAN)と呼びます．Intel 系のプロセッサはリトルエンディアン，Motorola 系のプロセッサはビッグエンディアンのため，PC から送信する際に，データの上位バイトと下位バイトの入れ替えの必要が生じる場合があります．

iMCs04 は LITTLE ENDIAN 形式でデータを扱っているため，Motorola 系のプロセッサを持つコンピュータと接続する場合，データの入れ替えが必要です．例えば，制御データ(ccmd 型構造体)の duty に 0x1380 を代入する場合，以下のような記述になります．


```
/*************************************************************
  unsigned char duty_num = 0x1388;
#if __BYTE_ORDER == __LITTLE_ENDIAN
  obuf.duty[0] = duty_num;
#else
  obuf.duty[0] = (0xff & duty_num)<<8 | (0xff00 & duty_num)>>8;
#endif
*************************************************************/
```

ただし，使用するPCが，どちらか一方に決まっている場合は，どちらか一方を記述するだけで正しく処理されます.

４.　ディップスイッチの設定

iMCs04 上のディップスイッチの Pin1～5 の操作により iMCs04 に固有の ID 番号を振ることが出来ます．全てのピンを 0 とすることで ID は 0x00（0）になり，全てのピンを 1 にすることで ID は 0x1f（31）になります．

ID	Pin				
	1	2	3	4	5
0	0	0	0	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	1	1	0	0	0
4	0	0	1	0	0
⋮	⋮	⋮	⋮	⋮	⋮
30	0	1	1	1	1
31	1	1	1	1	1

５．プログラムの実行

本章では，LINUX（Kernel 2.4 以上）で iMCs04 を操作する方法を説明します．以下の操作は全て root 権限で行ってください．

```
> su -l
```

（※ Ubuntu 等で行う場合は下記の様行ってください.）

```
> sudo (*コマンド*)
```

５．１　USBデバイスの登録

USB デバイスを登録します．本操作は，各 PC において最初の 1 回だけ行います．
コマンドライン上で以下のように入力してください．

```
> mknod /dev/urc0 c 180 140
> chmod 666 /dev/urc0
```

５．２　USBドライバのロード

ファイルがあるディレクトリに移動し，USB のモジュールをロードします．　まだ iMCs04 を USB ポートに接続しないでください．

```
> cd /home/user1/iMCs04/driver/
> insmod urbtc.o
```

（※ Kernel 2.6 用のドライバを使用する場合，下記のようになります.）

```
> insmod urc.ko
```

ここで，正常にモジュールがロードされているかを確認します．

```
> lsmod
```

と入力し，

```
Module                  Size  Used by
urbtc                   7360    0  (unused)
```

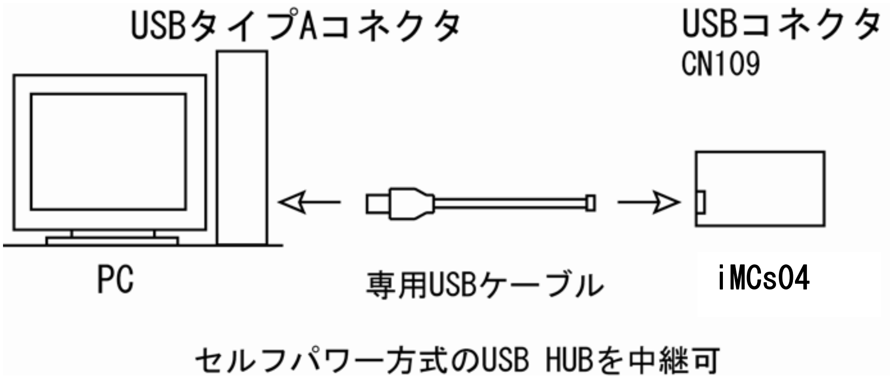
と表示されることを確認してください．

（※ Kernel 2.6 用のドライバを使用する場合，下記のようになります.）

```
Module                  Size  Used by
urc                     5557    0
```


５．３　ボードの接続

ボードを接続します．



接続後，

```
> dmesg
```

と入力し，

```
usb.c: registered new driver urbtc
urbtc.c: H8 based USB motor controller driver v0.1
hub.c: USB new device connect on bus1/2, assigned device number 2
urbtc.c: USB robot controller now attached to urbtc0
```

と表示されることを確認してください．また，HUB を中継する場合は，HUB の接続後（HUB には iMCs04 をまだ接続しないでください），

```
> dmesg
```

と入力し，

```
ub.c: USB new device connect on bus1/2, assigned device number 3
hub.c: USB hub found
hub.c: 4 ports detected
```

と表示されることを確認してください．

次に iMCs04 が正しく認識されているかを確認するために，

```
> ./h8test
```

と入力し，

```
Vendor xxxxxxxx
Product xxxxxxxx
```

と表示されることを確認してください．

５．４　終了処理

最後に，iMCs04 を USB ポートから取り外す際，dmesg で，

```
usb.c: USB disconnect on device 2
urbtc.c: urbtc0 now disconnected
```

と表示されることを確認してください．その後，

```
> rmmod urbtc
```

と入力し，USB ドライバを解放します．

（※　Kernel 2.6 用ドライバを使用する場合，下記のように入力します．）

```
> rmmod urc.ko
```

５．５　プログラムの実行

５．５．１　センサ値の取得

プログラムを実行します．サンプルで添付されているセンサ読込みプログラム uread を実行する場合は,

```
> ./uread
```

と入力します.

```
CH0:0xFFC0	CH1:0xFFC0	CH2:0xFFC0	CH3:0xFFC0
CH0:0xFFC0	CH1:0xFFC0	CH2:0xFFC0	CH3:0xFFC0
CH0:0xFFC0	CH1:0xFFC0	CH2:0xFFC0	CH3:0xFFC0
```

のような値が表示されます.

左から，buf.ad[0]，buf.ad[1]，buf.ad[2]，buf.ad[3]を表しています.

５．５．２　RCサーボモータの制御

プログラムを実行します．サンプルで添付されている RC サーボ出力プログラム sample を実行する場合は,

```
> make sample
> ./sample
```

と入力します.

５．５．３　RCサーボモータの制御（iMCs04 の複数台同時接続）

2 台目以降のコントローラを接続する場合は，以下のように接続する数だけ USB ドライバを追加します．本操作は，各 PC において最初の 1 回だけ行います.

```
> mknod /dev/urc1 c 180 141
> chmod 666 /dev/urc1
```

ここで，180 は USB ドライバのメジャーNo，141 はマイナーNo です.

3 台同時接続の場合は以下のようになります.

```
> mknod /dev/urc1 c 180 141
> chmod 666 /dev/urc1
> mknod /dev/urc2 c 180 142
> chmod 666 /dev/urc2
```

続いてプログラム（2 台接続時）を実行します．サンプルで添付されている 2 台同時接続 RC サーボモータ制御プログラム samplem を実行する場合は,

```
> make samplem
> ./samplem
```

と入力します.

改訂履歴

2002 年 10 月	初版
2003 年 1 月	Ver1.1 ・ピンファンクション訂正 ・ドライバ名訂正 ・ドライバのマイナー番号訂正
2009 年 9 月	Ver1.2 ・表記ミス訂正 ・パケット構造訂正 ・住所変更
2010 年 4 月	Ver1.3 ・ハードウェア部削除
2011 年 1 月	Ver1.4 ・Kernel 2.6 用のコマンドを追加

お問合せ(お問い合わせはメールにてお願いいたします)

株式会社イクシスリサーチ
E-mail : info@ixs.co.jp

本社所在地
〒212-0055
神奈川県川崎市幸区南加瀬 5-18-16

横浜工場
〒230-0051
神奈川県横浜市港北区箕輪町 2-12-29

この取扱説明書は, 再生紙を使用しています.